Intentional processing as a key for rational behaviour through Natural Interaction

Javier Calle-Gómez [a,*], Ana García-Serrano [b], Paloma Martínez [a]

[a] *Advanced Databases Group, Computer Science Department, Univ. Carlos III de Madrid, Spain*
[b] *Artificial Intelligence Department, Technical University of Madrid, Spain*

Abstract

This paper presents an interaction model pursuing flexible and coherent human–computer interaction. Starting from a cognitive architecture for Natural Interaction, an agent-based design is presented, focusing particularly on the role of the interaction agent. Regarding the intentional processing within this agent, the Threads Model is proposed. Finally, its implementation is described and evaluated to find out the integrity of the intentional approach.

Keywords: Natural Interaction; Intentional processing; Model-based design; Agent-based architecture; Threads Model; Task Model; Interaction system evaluation

1. Introduction

The use of language is a key tool for Natural Interaction, the research into systems interacting in the same way as humans interact with one another (Bernsen,

* Corresponding author. Tel.: +34 91 624 9117; fax: +34 91 624 9129.
E-mail addresses: fcalle@inf.uc3m.es (J. Calle-Gómez), agarcia@dia.fi.upm.es (A. García-Serrano), pmf@inf.uc3m.es (P. Martínez).

2000). This is far from a trivial statement, and all the issues underlying this claim should be examined. First, the potential of the language is more than bare 'information exchange' as is often assumed. Furthermore, the amount of information exchanged is often greater than the information acquired (by the receiver), with the quantity of accurately interpreted information being even smaller.

Although the fundamental and probably most important function of language is the information exchange, it is not the only one. Language can also be seen as a support for activities (like an act) and as some socio-cultural affiliation, apart from being a vehicle for exchanging emotions. As this tool is going to be shared between two participants in a two-way interaction, there is the risk of misinterpreting the set of features of language attributed by any of them.

Therefore, some 'bounds' on each facet of language use, and even certain bounds on the bare interaction, should be observed. Here, language is to be considered as the means for making the interaction possible in a restricted task-oriented domain. Hence, the first thing to do to characterise the interaction is to define and restrict the domain so that both parts in the conversation can share an 'interaction domain' or situation with the set of assumptions that it implies.

Looking at the situation, several interrelated aspects of which it is made up should be examined (Gee, 1999):

- *language* fulfils the semiotic plane in which sets of symbols, their restrictions, meanings and uses are observed;
- the *activity* plane, standing for the social activity in which the participants of the interaction are engaged;
- the *material* aspect, which concerns the environment of the interaction, consisting of place, time, participants (direct and indirect), etc.;
- the *political* aspect, which stands for the distribution of the social goods between participants, that is, social status, role in the activity, rank, etc.;
- and the *socio-cultural* aspect, involving certain knowledge relevant in the interaction related to matters such as social or cultural identities, values, etc.

Some assumptions about the situation should be met by analysing the nature and means of the interaction through discourse analysis. The framed interaction domain will include a set of rules determining the evolution of any interaction occurrence, just as the rules of a game determine the feasible moves each player can make at any time. The more restricted the rules, the fewer different moves available and the shorter the games, and consequently, the smaller the set of different matches. Of course, the set of actions is not truly 'restricted' from a Natural Interaction point of view, but the most frequent ones could be specified and the others generalised.

At this stage, frequent interaction occurrences should be analysed to get a model of the interaction domain based on the observation of its semantics, pragmatics, structures, intentional features, etc. Most interaction models rely on these observations through an interaction sampling. This task of 'corpus analysis' is a key point for attaining good coverage of the interaction domain. Corpora acquisition and

selection, which precede this analysis, have to look at the diversity of previously analysed situations to get enough corpora for a thorough analysis. It is, of course, no use acquiring a redundant corpus, but it is much worse to miss significant corpus.

The interaction model presented in this paper is a corpus-based one. The challenge is to apply intentional processing for attaining flexible, coherent, and successful interactions with little corpus. Yet restricted by an interaction domain, modelling this sort of human interaction knowledge will make system's behaviour more rational, approaching the Natural Interaction concept. It should be noted that the situations that this model is going to deal with will be between just two participants who intervene in turn, that is, bipartite interactions consisting of non-overlapped interventions. Though the model appears to be valid for other situations, this is left for further research.

This interaction model has been tested within the VIP-ADVISOR[1] project, with remarkable results which are detailed in Sections 5 and 6, including a processing example and an evaluation of the prototypes. Sections 2 and 3 will introduce the cognitive and functional approach. Section 4 introduces the intentional approach, as the core of the proposed model. Finally, Section 7 presents some related work.

2. Foundations of Natural Interaction

Throughout this section, we will explain the process of (natural) interaction, which will be broken down into a set of abilities whose accomplishment up to a point will lead to an approach to a Natural Interaction system architecture (Garcia-Serrano and Calle-Gómez, 2002).

Focusing cognitive aspects, a first break down will establish three major subtasks often found in classic systems of this kind: interpretation, application (or behaviour), and generation.

2.1. Interpretation phase

The acquisition and interpretation phase starts with the receiving of as much of the interlocutor intervention as possible, and ends with the comprehension and storage of any information found to be useful in it. Each intervention could include several discourses, which are sub-units with an independent semantic interpretation. These discourses could even be found through different modalities. Therefore, they first need to be differentiated, processed separately as required, and, finally, coordinated. The point at which they seem to converge is the semantic interpretation. Hence, it would be of use to find a common representation to bring them together easily.

Hence, any discourse should be interpreted (as its modality and language require) as a common semantic representation, which, together, is going to be interpreted at a

[1] VIP-ADVISOR project IST-2001-32440.

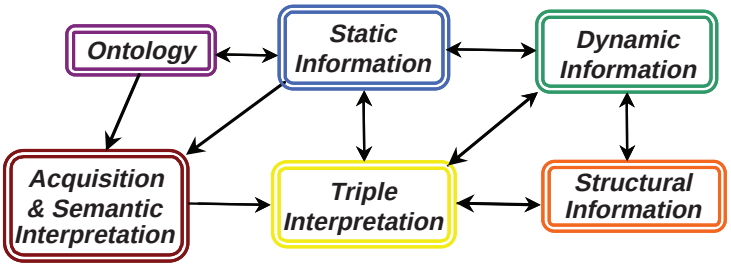

Fig. 1. Interpretation phase cognitive components.

deeper level. This interpretation should be looked at from three viewpoints: intentional, contextual, and structural, as detailed in Fig. 1.

The pursued interaction model is based on processing the direction and motivation of the dialogue (dynamic semantics). This will be called intentional processing and will be based on the interpretation and later management of *dynamic information*. But, as stated above, there is more information playing key roles within this model.

The second class is *static information*, used as the information covering the overall interaction by providing the semantic content details. This class is divided into two subclasses of information that are handled differently: data only applicable to the ongoing interaction, and data valid throughout several sessions. The first will be related to the session and handled by the session model. The main problems will be related to the *reference* and the *deixis phenomena*. The other subclass will be handled by the user model, whose main goal should be to recognise or categorize the interlocutor and then infer required information about him or her. From the acquisition to this point, the problem of relating concepts to the terms uttered will also appear (even through different modalities, as stated above). Therefore, we will look at the problem of conceptualisation, often left to ontology.

Finally, the third interpretation will analyse discourses searching for *dialogue steps* (structural information). Thus, sequences of interactions are handled through corpus-based dialogue structures (dialogue games).

2.2. Application phase

Through this phase, external processes should be attained as required. The interlocutor's last intervention has already been acquired and interpreted, and it is time to find out whether the interaction progress requires the performance of any task, unrelated to interaction management but concerning the *perlocution* (effect) of user interventions.

The sorts of tasks involved are sometimes knowledge-based, sometimes isolated functions of an application enclosed in the interaction management system. Either way, the interaction has to know which of the available tasks is needed, what its required entries are, and how to interpret the outputs, to feed them back into the dialogue. To facilitate this process, the tasks have to be organised in a Task Model, as detailed in Fig. 2.

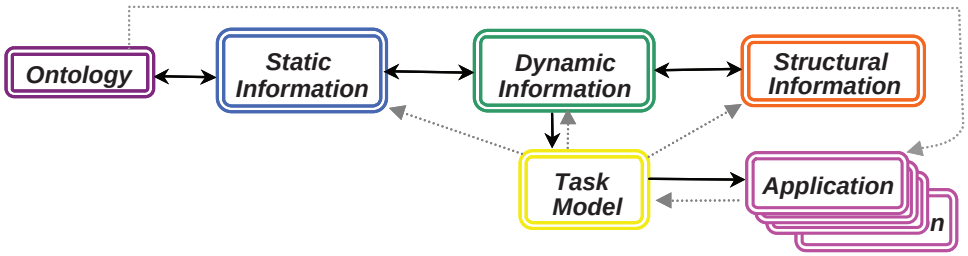

Fig. 2. Application phase cognitive components.

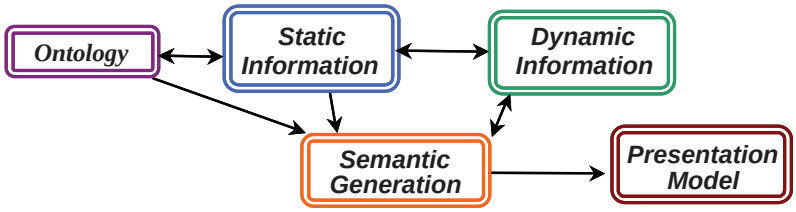

Fig. 3. Generation phase cognitive components.

The Task Model will determine which task, if any, has to be carried out in a given circumstance. The result of the task will be introduced into the dialogue, involving static, dynamic, and/or structural information. After introducing the result, another task may be necessary, so the task model will be revisited until there is no task left to be carried out for the given circumstance.

2.3. Generation phase

Finally, the last phase should finish off the interaction management with a semantic representation of the intervention that the system is going to carry out. This semantic representation will be input into a final process, the expression of system intervention.

The Presentation Model is in charge of this process, as shown in Fig. 3. It will have a set of modalities and languages, adapted to the interaction domain, the user, and the available media. It will assemble a coordinated output that will be presented to the user to conclude its turn and make the system ready for the user's next intervention. Then, the whole process will be repeated again and again until the end of the interaction.

3. Agent-based design implementation

In this section, we will look at another issue of Natural Interaction. Here the focus is not on cognitive questions, but on operational issues to approach an implementation. With this premise, three major subtasks were identified: the interface-related tasks, dialogue management, and the generation of intelligent content. This proposal will be specified through an agent-based system featuring three specialised

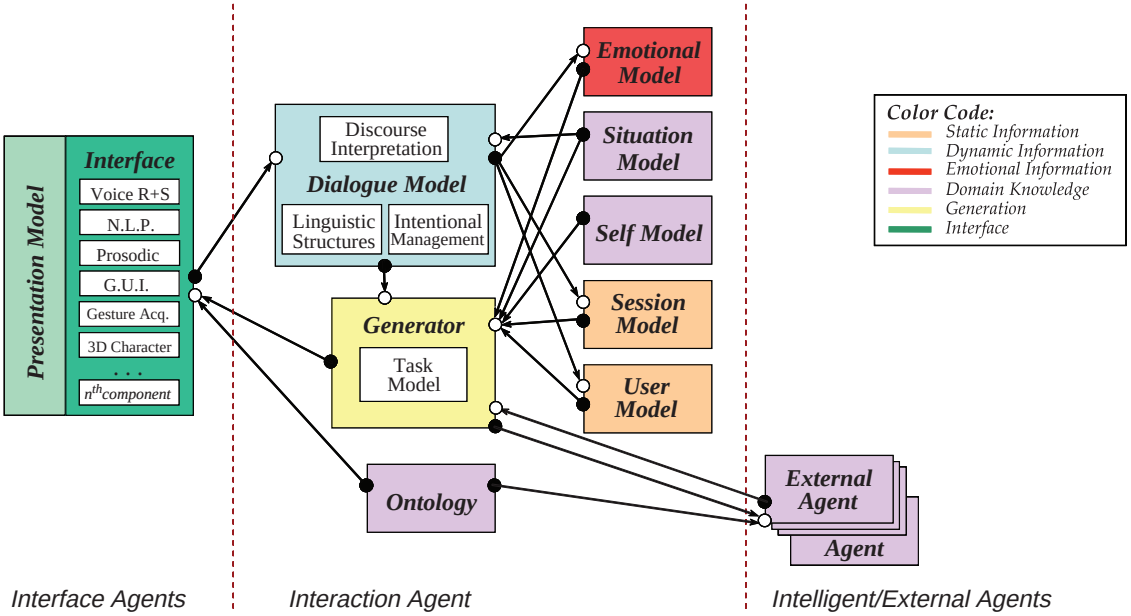

Fig. 4. Agent-Based Interaction System.

agents: the Interface Agent, the Interaction Agent, and a set of External Agents that carry out domain specific tasks. Their components and relationships are shown in Fig. 4.

The individual functionality of each of these agents is described below:

(i) the *interface agent*, in charge of interpreting every user expression and the generation of system utterances;
(ii) the *interaction agent*, whose role is to ensure that the dialogue is coherent and useful, specifically, keeping the context, managing the sequences of interventions, processing the intentional content, building conjectures about the dialogue state, etc.;
(iii) the *intelligent* (*or external*) *agents*, whose main tasks are to carry out actions (effect of the dialogue) and to provide knowledge-based content (based on the circumstances) when required so that the system is able to bring new content (even initiatives) to the dialogue and, hence, carry out *intelligent* interventions.

A communication language is needed to coordinate the different agents. In the first place, the communication between interface and interaction agents is based on a set of semantic structures that represent the (semantic) content of what the system has acquired from the user utterances. This proposal is centred on a set of communicative acts (Austin, 1962; Martínez et al., 2001), which will be divided into the six categories listed below:

- *representative*: when the emitter shows a link with the reality, aiming to share it with his/her interlocutor;
- *directive*, the emitter should direct subsequent actions to his/her interlocutors;
- *compromisive*, the emitter authorizes or commits himself with a line of action;

- *courtesy*, social conventions that both interlocutors should observe for reasons of protocol, tuning, or politeness;
- *non-verbal*, required for non-verbal social conventions such as connection;
- *null*, when there is no effective communication but an intervention is implied (such as content-empty discourses and pauses).

Several parameters characterise the occurrence of each of these structures in a way that can be shown as 'labelled' communicative acts to be handled by both interface and interaction agents. The communication between interaction agent and external agents would be either tailored to the applications or based on any general agent communication language, such as (FIFA, 2000 2002).

Going back to the description of the architecture in Fig. 4, it should be observed that the process begins with the interface agent. During the user's turn in a dialogue, the system will be awaiting the user's utterances. Then, these expressions will be interpreted as one or several streams of communicative acts that will be later delivered to the interaction agent. The structures will be exactly the same, no matter what their source was (hence, their expression modality). The interaction manager will interpret them and then pass them on to the Task Model and the external agents. After they have been carried out, and their result has been interpreted by the interaction agent, system intervention will be generated and represented by one or more streams of communicative acts (one for each discourse that the system is going to carry out). These communicative acts will, finally, be received and expressed by the interface agent. To give a better understanding of these agents and their participation in the interaction, the following sections include a brief description of each.

3.1. The interface agent: acquiring, interpreting and expressing

As stated above, this agent's function is to acquire and interpret the interventions of the interlocutor (from now on referred to as the user, whether he or she be human or not), as well as expressing system interventions. Generally, there will be several ways open to the user to make his intervention. The VIP Advisor project looks at two of them: verbal expression (either spoken German or written German or English) and the graphic user interface. No matter which way the user chooses, these interventions are going to be interpreted and translated into the same semantic structures: communicative acts.

As regards the output, this agent is going to receive the same sort of structures, and will make use of three ways of expressing them: verbal expression, GUI, and the 3D avatar, which will complement each other to express jointly what the agent is asked to do. It will have a Presentation Model for distributing the contents to the proper component.

3.2. The interaction agent: achieving a coherent flexible Dialogue Model

The static information within a session is going to be divided into atomic pieces according to the representative part of the speech. The representative acts (inform, confirm), and any other act implying static information, will be processed and that static information stored.

Regarding the organisation of these pieces, it will follow the dialogue organisation. The interventions are organised as dialogues and subdialogues, hence the static information should be linked to the subdialogue in which it appears. This information is characterised by its credibility which will be higher as the distance between subdialogues becomes shorter. Sometimes, a piece is used out of the dialogue in which it appears, as a result of inheritance processes. In general, within a subdialogue, the context of precedent major dialogue is also applicable (but perhaps with less credibility). What is more, when a subdialogue comes to an end, some of its static information is still valid for the precedent subdialogue. Furthermore, this also applies even when a dialogue simply loses the focus (that is, the dialogue is not ended, but another subdialogue is opened or reopened somewhere else, hence the first one loses the focus).

Each piece of static information is going to be characterised with a certainty value, that is, an indicator measuring how sure the system is about the accuracy of the piece. Thresholds for some pieces of information (or classes of pieces) could be defined so the session model can evaluate whether the information to be provided is good enough for its purpose. If not, some reinforcement strategy may be applied, which could end up with a new subdialogue. For example, the system can propose the value and then ask for confirmation or even just mention and use the information found.

Static information reinforcement might also be used as an indirect way of asking. Let us take a dialogue state where a default value for some feature is introduced into the context as a presupposition, but with a low certainty value. When the system needs to use this value, it will find that a strong reinforcement is needed. So, by applying a reinforcement strategy, the system could be forced to ask the user for that value, but the sort of subdialogue will be different because it already has some information (as explained above).

There is a special sort of static information that is not directly linked to the intervention in which it appeared, but to the actual interlocutor. This information has another important feature: it is valid throughout different sessions. For this kind of static information another model should be observed: the user model. In this model, the identity of the user is a continuous incognita and will be processed throughout the whole session. Every time some information regarding the user is acquired, it will be used to restrict the concept of user. When some information about the user is needed, it will be restored or induced in a way that will influence the whole interaction process. This sort of User Model stores links between values for the significant user features observed during the system's life, throughout all sessions held with different kinds of users. The user is unknown when starting a new session, but he or she will match a 'class' or group of users anytime (average user at first). Hence, when some information about the user is needed, the feasible values for that feature would

be bounded, and there could even be a predominant one. These values should also be proposed with a certainty value, which is liable to be reinforced as in the aforementioned in case. However, this sort of model involves a high cost and does not significantly affect the intentional processing, which is the real core of this work. Hence, this model is not included in the prototype (which will be referred and evaluated in Section 6), and the information it should handle will be managed by the session model (like any other static information). Anyway, it should be stated that overall performance could be improved, in general, by including this sort of model.

Regarding intentional processing, it could be stated that during any communicative process each participant will be located on a plane of interpretation. For a successful interaction, they have to be on the same (or at least a similar) plane. When communicating, people often check the plane of their interlocutor and negotiate a 'common ground' to make interaction useful. When both of them agree on the same plane, it can be said that they have reached a commitment (Cohen and Levesque, 1991; Clark, 1996). These commitments should not only observe the information on each participant's aims and agreements, but also what the other participant knows about the other's aims and agreements.

As mentioned above, *dynamic information* concerns the underlying intentions during the dialogue. Within any subdialogue, a discourse line or 'thread' could be identified that carries the intentional information of that dialogue. Thread organisation and processing will give rise to intentional processing, which is the key to the present model and will be discussed at length later.

Finally, the 'discourse maker' (see Fig. 1) has to implement the effects implied by the development of the current thread, if any. Some interventions, such as protocol greeting for example, will not imply any effect. However, certain effects will often need to be achieved by invoking a given functionality. Every functionality liable to be invoked will be called a 'task'. Furthermore, all the tasks should be organised in a way that facilitates the recognition of which task is necessary (if any) at any time during a dialogue before generating a discourse.

These tasks are usually *knowledge-based tasks* (*KBT*). However they could be any task, from some direct change in the state of system (either state of the interaction or any of the system properties, such as environment variables) to some function requesting any system application (which could mean some command to external devices, such as the printer, for example). These tasks will be often classified as 'external' or 'internal', but there will be other less general criteria such as 'who is in charge of carrying out the task'. Therefore, it will rely on a set of external agents able to carry out tasks, as well as a Task Model Module able to organise their participation and carry out internal actions in dialogues.

Modelling the tasks will once again mean the analysis of the interaction domain corpus to identify the tasks involved. This corpus analysis will be integrated into the discourse analysis for the interaction agent. This means that every task will be located within a set of circumstances of the discourse, thereby identifying a task needed at any time within a dialogue.

But this is not the only information embedded within the task model. It will also be necessary to see how to carry out the task (how to ask for it, entering its inputs

and collecting its outputs). And, of course, it will be essential to record how to interpret the result (the task outputs) to be fed back into the dialogue as required. Note that interpreting the result could either mean new static information, event triggering (which could mean changes in the game state of the current thread), or even introducing new initiatives. Hence, as a result of the task performance, the system will be capable of proactive behaviour.

When the task has been carried out, it will again be necessary to analyse the circumstance through the Task Model because the task performance will have changed it. Then, another task will be chosen, until the dialogue needs no more effects to be attained but the system illocutionary acts. At this point, the discourse generator will have to look for an appropriate pattern to express what it has found during the development of the thread. These 'illocutionary patterns' will also be discovered through the analysis of the interaction domain corpus and ought to be recorded and organised within the discourse maker, the last component of the discourse generator.

4. A proposal for intentional processing: the Threads Model

Throughout the previous sections, we have presented a cognitive architecture for interaction as a frame for a dialogue model based on intentional processing. We have already stated that the intentional processing in this interaction model is going to be discharged by a model that is based upon the notion of thread: the Threads Model (Calle, 2004).

Answering to the name 'threads' will be the lines of discourse underlying any intervention (and, more generally, the entire dialogue). Each thread should be identified as a minor interaction unit, a sub-dialogue intended to attain a goal (or, more generally, a set of goals). These units could be seen as dynamic (pragmatic) information containers, whose existence should be discovered by analysing the interaction domain corpora. Though these units could be generalised into a set of 'universals' (as well as other related linguistic units), which repeatedly occur throughout different interaction domains, it is of no use trying to implement all of them but only the sort of threads involved in the analysed interaction domain. Hence, the threads will be properly identified and characterised during the dialogue corpus analysis for their later implementation (within interaction system development).

It has been said that the threads stand for and carry the intentional information of any dialogue. Just as these intentions (discourse lines) concerning a given interaction are related to each other, the threads depicting them should also be inter-related. Thus, the threads will be organised as they appear and develop during the dialogue, showing its intentional structure.

4.1. Thread structure

As with many other data of all kinds, the human mind usually classifies the intentions underlying any interaction in hierarchy-like structures (Grosz and Sidner, 1986). There will be general dialogues that will require minor sub-dialogues for their

accomplishment. Each sub-dialogue could also rely on its own set of sub-dialogues for its development. There will also be 'dead-end' sub-dialogues, related to no other dialogue except the major precedent one. Finally, an 'overall' unit standing for the entire interaction should be considered, whose development means, of course, a set of major dialogues. In terms of intentional process, this overall unit represents the connecting intention, that is, the aim of entering into conversation with the interlocutor.

This latter unit is going to be called the *base thread*, and is the only thread that has no precedence relationship with any other thread (no precedent threads). At the same time, any thread could give rise to new minor threads (none, one or more), which have accurate intentional interpretations. Thus, it could be concluded that dialogues can be organised with the help of a tree structure.

4.2. Thread structure of static information organisation

As mentioned above, this intentional structure will determine the contextual (static) information as well. Each thread is going to be characterised with a context line. When some static information is needed for developing a given thread, the session model is going to be asked for it.

The session model could possibly have the information required related to the given thread, and it will meet the request by just providing it. If not, it will have to search all its thread ancestors for the information. If the data is found, it will be inherited but its certainty value will be decreased by a percentage proportional to the distance from the given thread to the one which owns the information. This sort of information transfer will be designated as top-down inheritance.

There is another kind of information inheritance between threads that should be pointed out: bottom-up inheritance. For this to occur, some data of the thread context needs to be characterised as 'relevant', either when creating the thread or when the information is acquired. When that thread loses the focus, it will leave its relevant information in its parent's context, of course, with some loss of certainty.

4.3. Attentional structure and commitment

There is another information structure, the *focus*, for fixing attention onto the proper node of the tree at any time. It mainly consists of a stack of thread node identifiers. As the threads are proposed and committed, they are put into focus for a normal nested dialogue. When a thread is solved (or better 'faded'), it will be taken out of the focus. Therefore, a previously focused thread remains on top of the stack and will be the most likely candidate to be developed next. However, the user may or may not 'jump' to any thread anywhere in the focus structure at any time, and even jump to a thread out of it (faded). Hence, intentional interpretation has to be prepared to receive out-of-the-focus interventions.

Therefore, focus just points out a good candidate to be the 'current thread', that is, the thread supposed to be followed by the user through the interpretation of his or her intervention and by the system during the generation of its intervention. But

it says nothing about either interlocutor being prepared do develop the thread. They could have little confidence in the success of the thread if they do not have all the information about it, or worse, they could even be unaware of the existence of the thread. Hence, when developing a thread, each participant needs be confident that his or her interlocutor is prepared to cooperate in this activity (that is, the interlocutor knows that the focused thread exists, and he or she has enough information about it to have it developed). All these matters will be summarized in the commitment, a feature to be added to the thread definition.

Consequently, a gauge of commitment will be added to each thread to measure the system's belief about its commitment. Every time a thread is processed, it will have its commitment watched, and if it is found to be too low, it should be raised using some reinforcement strategy. If this is not possible, the focus will have to change (to an existent prior, or to a newly proposed, thread). This feature, the commitment management, is part of the 'common ground' modelling in the Threads Model, and one of its strongest points.

In addition, when a thread is faded and put out of the focus, the next thread could have been developed elsewhere (resulting from some jumps), and its commitment may be so low for it to be skipped. For an accurate processing of these sorts of behaviours, the focus review has to have an auxiliary structure called 'focus history', which consists of a list of the threads processed throughout the whole session.

Finally, a last issue on focus management should be noted. A set of domain dependent rules could be included in order to prioritize certain kind of threads. Thus, the system will also be capable of 'jumping' to an unfocused thread just because the system finds its development more profitable than the one focused on. However, this sort of behaviour may be considered aggressive by the interlocutor and should be avoided.

4.4. Threads model components

The model designed includes three components:

- the user thread: regarding user intentions, as they are acquired and interpreted;
- the system thread: how the system introduces new threads when required for other thread development or when any component of the interaction architecture demands it;
- the thread joint: depicting that common ground between user and system intentions.

When carrying out system interventions, the thread joint is the lead structure, trying to bring both participants onto the same plane of interpretation. The other two are useful for enhancing interpretation and task performance.

Summarizing, thread modelling provides a mechanism for organising, interpreting and planning dialogues, following the intentions shown by the user interventions and the system's own initiatives, raised either from a thread grammar or motivated by the result of any component in the dialogue. This organisation affects the arrangement of the static information (context).

4.5. Example of thread implementation (for a specific domain)

Within the specific interaction domain of the VIP-Advisor project, several threads were found and implemented. A brief description of the main threads is given below:

1. *base thread*: aims to set up interaction for both interlocutors;
2. *command*: tries to make the interlocutor carry out some action;
3. *request*: aims to force the interlocutor to provide some (static) information;
4. *empower*: offers, recommends, or makes the interlocutor propose some initiative (a particular one, or any initiative in general). It is often used by an interlocutor when he or she wants a given thread to be developed but is not supposed to initiate (or retake) this thread, making it clear to his or her interlocutor that this is the line to be taken;
5. *solve*: closing thread which intends to finish off an existent thread or to inform that it has already ended with a result.

The base thread is proposed at the beginning of the interaction. During its development, other threads could be added. Any time a participant makes a discourse, it could follow an existent or initiate a new thread. New threads should be of the 'initiative' type (that is, any one but a 'solve thread'). While developing a thread, it could reach a closing state in which the focus will be lost. Then, any other thread could be chosen for development and so on. The focus structure will always contain at least one thread, the base thread, so there is always a solution for the next thread selection. When the base thread is developed to the end, it will mean that the intention of holding a dialogue between both interlocutors is over, and so is the whole interaction.

Apart from these, there are some other minor, yet very important, threads, such as the *confusion thread*, proposed by any interlocutor who detects that the current thread is in danger of failing resulting from some misinterpretation at any level (either semantic, intentional, etc.). Its aim is to introduce a strategy to minimise the risk of failure, which could range from informing the interlocutor of the event, up to improving interpretation. This will depend on the criticality, its resources (perhaps some auxiliary interpretation agent), and, of course, the corpus for these lines of dialogue. It could even lead to another thread, which has been labelled *reinforcement thread* and whose aim is to raise the 'quality' of the doubtful information (either static or dynamic). Therefore, this thread should end up raising the context certainty or the commitment (if any) by applying some reinforcing strategy, such as announcing, reminding or direct asking. There will also be a thread for stating spontaneous dangers to the success of a thread that have nothing to do with the mutual interpretation of the participants, but with some external (environmental) events. Such threads are labelled *warnings* and should state the case and ensure that both interlocutors share that knowledge.

Finally, there is also a *waiting thread*, standing for a participant's intention to wait for his or her interlocutor to act after releasing the turn. It is a very special thread

because its progress is based not on communication but pauses. Any other act from the interlocutor will immediately end the waiting, but a sequence of pauses will make the not-in-turn part take the floor again to revive the dialogue. In such cases, depending on the corpus, the waiting party usually states that he or she has released the turn (reminding his interlocutor that he or she is waiting, for example), advances the progress of the current thread (prior to waiting), or even proposes a new initiative or changes the focus to another existent thread. If the thread to be advanced is the base thread, it could reach a state in which the waiting party says goodbye, and then another state in which this thread is complete. This will mean that the connection intention is over, and so is the whole interaction.

4.6. Thread development through interactions

We then have the problem of *developing* a thread. Apart from intentional processing, several pragmatic studies point out that there are predefined protocols and interchanges of elements that are commonly used for certain purposes. One of these is the organising principles (Levinson, 1983), to understand the dialogue as a shared co-operative social task in a way that the steps in the dialogue are the illocutionary elements that are carried out by the interlocutors in turn. Subsequently, there will be a set of patterns for each kind of dialogue, and any interaction should fit a pattern of its class. These patterns will be called 'scripts'. The user/system interventions could be seen as steps throughout the dialogue according to some particular set of rules called 'adjacency pairs'. This dialogue management is known throughout the literature as 'dialogue games' (Levin and Moore, 1977; Kowtko et al., 1993; Amores and Quesada, 2001). This approach could be seen as complementary to intentional management such as the Threads Model: each thread always has a state (see Fig. 5).

When introducing an initiative, a proper game has to be chosen, assigning its initial state to the thread. Any time a thread is inspected, its state will reveal how it is being developed, and where that development is located (even imminent steps to be taken). The State Manager that supports the thread management in the Dialogue Manager is implemented as an automaton which is also used before interpreting a user intervention, providing a prediction of the feasible paths the user can take on his or her own.

The transitions in the automaton may or may not be induced by user moves (normal transitions). When they arise with no explicit move by the user, they could be a result of three different sources, which lead to three kinds of motion:

- λ-transitions (ellipsis): postulate that the user has omitted a step in the script;

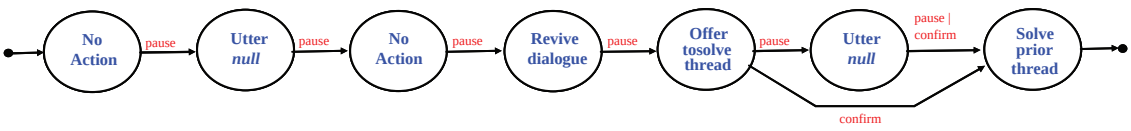

Fig. 5. Diagram of a *Dialogue Game* for the waiting thread.

- event-transitions (ε-transitions): resulting from some performance or task execution by the system (or any external agent);
- π-transitions: carried out by the system intervention to change the state during its intervention (which can also be seen as a particular case of ε-transitions);

Note that these games, with their states and transitions, should be acquired from analysing the dialogue corpus. Once the dialogues have been broken down and each subdialogue has been identified, they have to be analysed, classified and then implemented. Again corpus analysis plays the key role.

4.7. Dynamic games solving the lack of corpus

One of the most important problems for corpus-based interaction systems is what to do when corpus is found to be insufficient, and dialogue arrives at an unknown point. The user may ask a question that the system is simply unable to answer, or perhaps a more complex strategy is needed to solve the embarrassing situation.

Given that the interaction agent is also an agent, it could also be summoned by another interaction agent when this sort of problem is found. A new task will be added to the Task Model, so the interaction agent will ask for help when a problem is found. If the available interaction agents have modelled the same knowledge as the one making the request, this will get no more response than what it already knew, and another technique will be needed (such as apologising and closing the current subdialogue or introducing a new subdialogue to clarify what to do by cooperating with the user). But maybe there are other interaction agents available prepared for a different domain (with different corpus), or based on a different technology, or just with a great deal of corpus. Anyway, if they found an appropriate solution and passed it to the embarrassed interaction agent, it could make the most of it.

What the interaction agent has to do is to create a new thread, and (if it receives a solution from some external interaction agent) create a dialogue game associated with that thread which will carry out the solution proposed by that external interaction agent. Depending on the level of detail of the solution provided, it will have to resign itself to applying the solution (as a black-box solution) or could even learn it for future use (given all the task details and, of course, given that the user approves the solution by reinforcement in subsequent interventions).

In the VIP-Advisor project, a Case-Based Interaction Agent was summoned for these tasks with very satisfactory results (especially for question answering, concept definition, tutoring, etc.).

5. An example of thread management

This section will describe the example shown in Fig. 6. When the user connects, the system utters its first intervention (S1). It is a protocol greeting (with no other

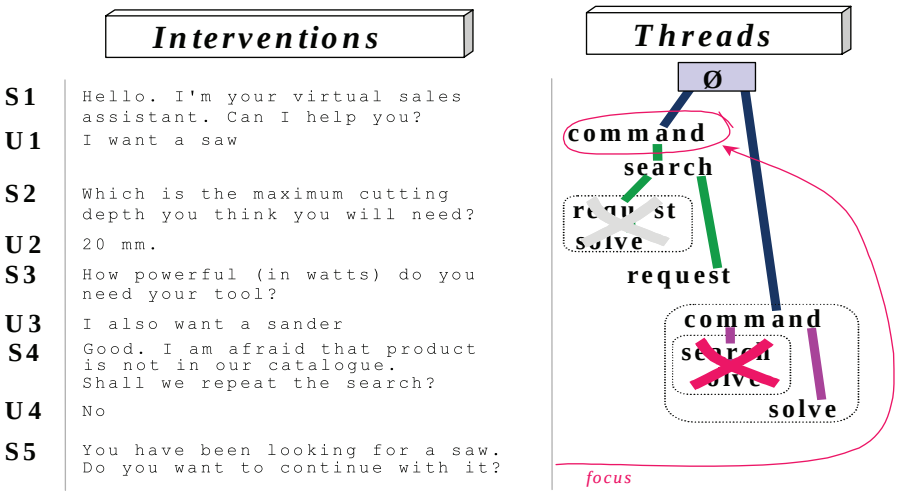

Fig. 6. An example of Thread Management.

intention than advancing the base thread). It then introduces a generic 'authorize' initiative to encourage the user to introduce his or her initiative. A rejection of this authorization will force the system to go on advancing the base thread, which will probably lead to a short farewell followed by the end of the session.

However, the user proposes his initiative (U1), consisting of the (product) search command. Before the next intervention (S2) can be attained, the system has to summon a knowledge-based intelligent agent to provide a response (in this case, product identification). The KBA finds that there are several feasible responses, and chooses a criterion to distinguish them (particularly, the desired cutting depth for the product). This criterion is input into the Task Model, which decides to create a new thread to get more information from the user in order to refine the solution. Since the entry point of the Discourse Generator changes (new focused thread), this component has to restart its process. This time, the Task Model has no need for external help in attaining its intervention, and chooses some patterns for expressing its desired discourse.

After the user answers the question (solving the request subdialogue), the Task Model again summons the KBA, this time with more information within the context. Unfortunately, the information is insufficient so another criterion is chosen, and the consequent request initiative is proposed by the system (S3). This time the user does not feel like answering at once and goes on to propose a new initiative (U3). It appears to be a new command for another search, and after the interaction process, the Task Model will probably ask the KBA for a solution. This agent finds no possible solution, and lets the interaction agent know. So, the interaction agent reports the solution and empowers a refinement (S4), following the chosen dialogue game for that command thread. The user rejects it (U4), and this way the thread reaches a final state, and is finally solved.

At this point, the system has to choose where to fix its attention. The focus will suggest it takes the unsatisfied request that was proposed last, but this thread is far from the last developed one. So it will have to take the lower common

ancestor of both of them (the solved and the newly focused thread). This time, the common ancestor is the base thread, so this will be the one to take on the focus. If it has enough commitment value, it will be progressed for which it is necessary to retake its successor (the command thread), and if its commitment value is high enough its successor will also be retaken, and so on. However, the commitment value of the first command thread is too low, and the 'thread joint manager' will seek to improve it by using some reinforcement strategy. In particular, it chooses to ask the user directly to commit the thread again (S5), which will reinforce the whole branch. A rejection of this authorization would have meant fading the thread and all its successors, taking them out of the focus and returning to the base thread for its development another way (which would probably lead to a generic authorization such as S1 or a farewell and disconnection). On the other hand, if the user chooses to approve, the thread will be reinforced and, having the focused thread ready to be developed, the system will go on.

Note that intelligent agents are not the only ones that can propose initiatives. In fact, almost every component in the interaction might need to introduce initiatives. For example, the session model could need to reinforce the credibility of a context piece (whenever the use of the piece determines a certain threshold, and the credibility is under the required minimum). Then, this component should introduce an initiative for reinforcing the 'quality' of the information (that will probably lead to a subdialogue starting with a question like "the value feature X is V, isn't it?"). Even the very thread management components could propose initiatives (as explained above, when the 'quality' of a thread, that is, its commitment, is under a predefined or user-dependent threshold).

6. The evaluation of a Natural Interaction system

There are two main perspectives for evaluating an interaction system: diagnosis evaluation, for detecting, classifying, and fixing interaction problems, such as linguistic, misinterpretation, etc., and performance evaluation, measuring the quality of the interaction achieved throughout a series of predefined scenarios in which certain aspects were focused and quantified. Though the rates and quantities are subject to interpretation, it is very reliable as a gauge for the enhancement of an interaction system (through the comparison of different versions of the system), or even as a balance for comparing different systems within the same domain (based on their main features).

This section will be aimed at measuring the advantages of modelling the 'common ground' and commitment handling through Natural Interaction, for which two different (near consecutive) versions of the implementation of the interaction model presented in the following sections will be compared: one with these features, and the other without them.

Thus, comparative performance evaluation will be focused on. For an accurate comparison, both systems should be tested by similar teams of users, ideally the same

team. Most interaction domains, apart from some trivial ones, will require large teams of users for approaching this concept of 'similarity'. On the other hand, the problem with getting it tested by the same team of users is that they might be influenced in their later tests by the first ones. For minimising the impact of test users training through first tests, a significant time gap between these and later tests should be set.

Next, we will inspect some aspects and indicators of relevance for Natural Interaction. Then, we will review the results of an evaluation by comparing the two versions of the interaction system through a set of sample scenarios played by a team of testers in two sessions, separated by a six-month gap.

6.1. Some indicators for performance evaluation

Throughout the following paragraphs, some relevant aspects of Natural Interaction will be explained to focus on performance evaluation, based on other works such as Bernsen et al. (1998). In addition, some indicators for each of those aspects will be proposed.

Goal proposal and achievement. The number of goals underlying the intentional structure of the dialogue will indicate both the complexity and the success of the interaction. In addition, it will also characterise the dialogue depending on the proportion of initiatives proposed by any of the participants (user guided, system directed, mixed initiative).

- number of user goals: number of total initiatives proposed by the user;
- number of system goals: number of system-directed goals;
- goals achieved: number of successful goals throughout the entire interaction;

Hold of commitment. Both the achievement of a goal and the good progress of the dialogue are indicators of the suitable process and hold of the commitment. Nevertheless, there are other particular gauges for this feature.

- useless interventions: number of non-null interventions that are unprofitable for the interaction;
- buggy interventions: number of non-null interventions resulting in some confusion, so the dialogue needs extra interventions to be restored (if it can be);
- recovery discourses: fragments or entire interventions used to restore the commitment and to settle confusing situations. As opposed to the above point, this is not a negative indicator, because recovery discourses can be used for prevention;

Dialogue progress. The quick progress and the profitable use of the dialogue reveal a healthy interaction and good tuning between user and system. For task-oriented interaction systems, we should comment that the results at this point should be better the more restricted the dialogue domain is (for its particular task).

- user steps through an intervention: average number of things the user achieves through his or her interventions in a dialogue;
- system steps through an intervention: the same as the above, but regarding system interventions;
- number of interventions from user and system: percentage of participation (turns taken) between user and system throughout their dialogue;
- number of null interventions: number of interventions meaning no progress (no change in any, either informational or operational, state).

6.2. Evaluating two consecutive versions of the same system

In this section, two similar versions of the same interaction system are going to be summarized and compared. Both of them follow the interaction approach proposed in these pages, and both were set for the same domain. The main difference between them is that the first one lacks commitment handling (hence the thread 'quality' is not assured) and, consequently, it has no reinforcement process. This evaluation should show the effect of these improvements in the interaction.

The interaction systems evaluated have, of course, an interaction agent, and the interface and external agents as described in Section 3: some *interface components* (Natural Language Processing, voice recognition and synthesis, graphical user interface, and a 3D character), coordinated by a presentation manager; and External Agents, for providing assessment and concept definitions in the Risk Management and Assurances domain. Regarding the interaction agent, it has an implementation of the Threads Dialogue Model, which is to be evaluated, both versions differing in the thread management. But it also requires and includes a Session Model, a Task Model, a plain Emotional Model, and an Ontology. Other needs (Situation, Self, and User Models) are carried out by the Session Model (as a context of the *base thread*, the root of the intentional structure). Both versions of the interaction system were implemented in Ciao Prolog (Ciao, 2004) (Dialogue Model and NLP components) and Java™ (Java Technology webpage, 1994 2006) (the others).

During these tests, 25 test dialogues were used for each version. They were performed by five testers, not involved in either the design or the development processes. Each tester had five scenarios, carefully chosen for being as different and representative as possible in the interaction domain. The dialogues produced should be framed as 'inside the domain, but not necessarily inside the corpus'. The scenarios and the testers were the same for testing both versions, but the tests were separated by six months mentioned previously.

As shown in Fig. 7, the goal proposal and achievement evaluation, several significant results have been obtained from version 1 to version 2. First of all, not every goal succeeded in version 1 (mainly due to misunderstandings and commitment losses). However, given the average number of turns carried out in a dialogue, it takes less than four turns to accomplish a goal (0.28 goals achieved in a turn, and 0.53 goals initiated in a turn). This gives an idea of the involvement of both system and user (tester) in the success of the dialogues. In the second version, this feature is maintained and even improved on (around 0.6 initiatives proposed each turn),

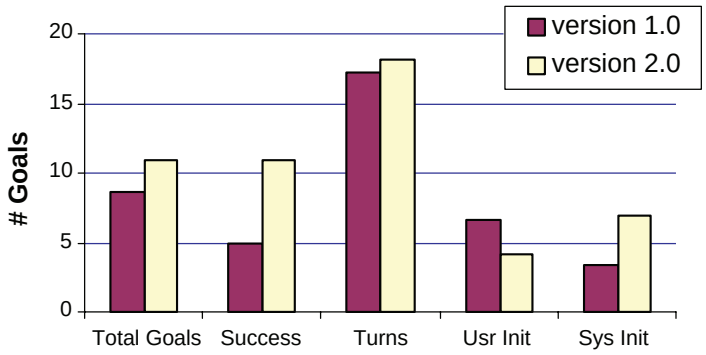

Fig. 7. Goal Proposal Evaluation Comparison.

apart from ensuring that almost every goal is properly recognised by the interlocutor (either the user or the system) and consequently solved (at least, at a linguistic level).

Although the domain was mainly 'system directed', version 1 implementation was not very aggressive (revealing just 3.44 average interruptions while the user interrupts an average of 6.56 times in a dialogue). This was a feature to be improved on and, as can be seen from Fig. 7, the second version follows the dialogue more closely anticipating its interlocutor (maintaining the commitment between them). Hence, the second version implementation is more aggressive, introducing more initiatives than the user, although retaining the mixed initiative interaction approach.

Regarding the commitment and utilisation evaluation, some final data for both versions is shown and compared in Fig. 8. We find that only 1% of total interventions in the dialogues held throughout the test for version 1 are buggy ones. Although this is not a lot, intentional commitment should be improved to rule out these situations as far as possible. The second version reduces this drawback to less than a third of what it was in version 1 (just 0.31% of total interventions).

As mentioned above, these buggy interventions cause the user and system not to be attuned to the same plane of interpretation, and the dialogue needs to be restored using 'recovery discourses'. Version 1 only uses recovery as a solution for misunder-

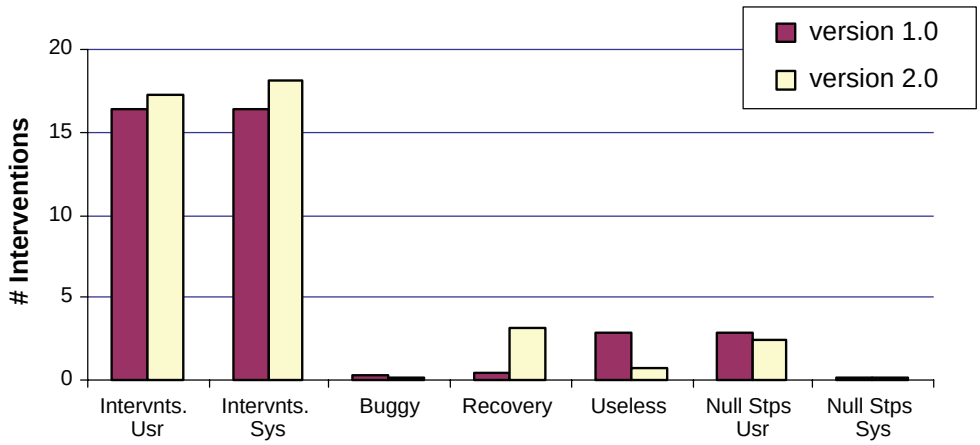

Fig. 8. Utilisation and Commitment Evaluation Comparison.

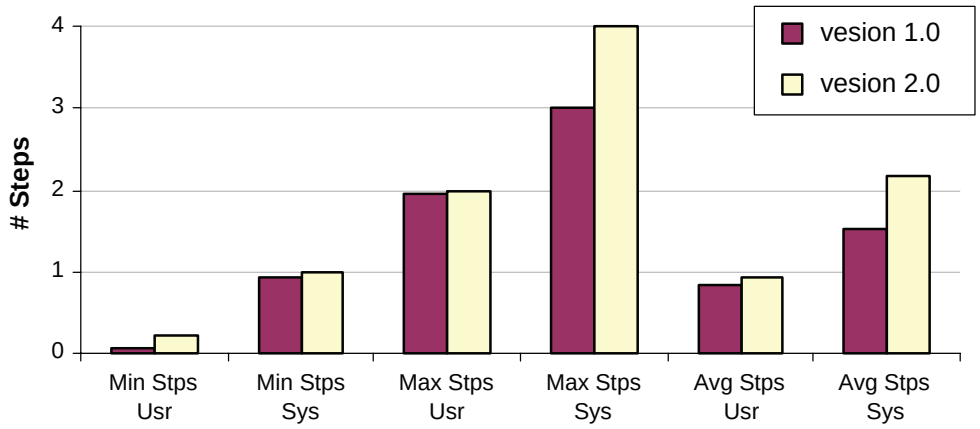

Fig. 9. Progress Evaluation Comparison.

standing, but it is never used for prevention or commitment handling. The recovery discourses involve 1.5% of total interventions in the dialogues for version 1 tests. Remarkably, this quantity happens to be increased throughout the tests for the second version (up to 17.2% of total interventions are entirely or partially involved in recovery actions). The reason is to be found in the preventive use of recovery applied in the second version (not just for misunderstandings, but also used when the commitment is found to be weakened somehow).

Preventive recovery, along with other processes designed to raise commitment, also appears to be the cause of a reduction in the number of buggy and useless interventions. As shown in Fig. 8, the version 1 tests reveal that nearly 9% of total interventions were of no use, while the second version reduces this percentage to 3.7%. These results should be interpreted as most of the useless interventions in version 1 being mainly due to the lack of tuning between both participants, which was significantly solved in version 2 with its preventive recovery and reinforcement of the common plane of interpretation.

We also see from Fig. 8 a slight reduction in the null-step interventions carried out by the user, which suggests that the overall use of the application is a bit easier or perhaps the testers were somehow more settled or accustomed to this sort of interaction system for the second evaluation.

Other interaction progress indicators are set out in Fig. 9. Note that, throughout the dialogues held with version 1, the average number of steps carried out through any intervention was 0.84 for the user and 1.51 for the system interventions. Although these results reveal the rapid progress of any dialogue, they have even been increased for both user and system in the second version (0.92 and 2.17, respectively). User progress approaching value '1' could also mean easy usage, but, more importantly, cause the system to try to go faster. The user is comfortable with the interaction even if the system progress is nearly 143% faster. Therefore, it appears to be profitable to introduce strategies that, although consuming linguistic resources, manage to achieve an overall acceleration of the interaction.

It should be noted that the final length of the interactions is slightly greater in the tests carried out for the second version (Fig. 8), which seems to contradict what we

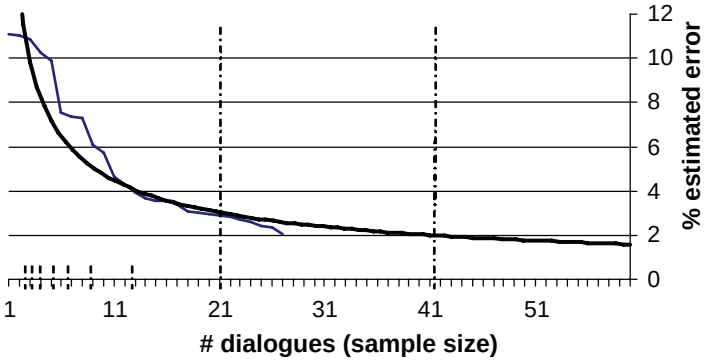

Fig. 10. Evaluation Accuracy.

said about quicker progress. However, we also need to take into account that the number of goals achieved is also higher, which clarifies the odd interaction lengths.

6.3. Evaluation summary and conclusions

The second version appears to improve the effectiveness of version 1, by achieving more goals and making quicker progress in any dialogue. In addition, the greater involvement of the system in the interaction success leads to easier usage and 'denser' interaction (more useful). Finally, preventive recovery and reinforcement of the common plane of interpretation appear as a key for attaining comfortable and profitable interaction.

Analysing the estimated error and the size of the sample, it has been found that 20 dialogues should be enough for reasonably good testing, as shown below in the graph (Fig. 10). For a precise testing of Natural Interaction systems in task-oriented domains, at least 135 tests should be carried out (but this, of course, depends on the interaction domain).

7. Related work

From language to the very intention of the speech, both speakers need to be primed for the best performance, since human speaking is a process of agreement. And they both try to do their best, since success in communication depends upon the efforts of the two of them. Hence, when communicating, a speaker is constantly looking for the plane of interpretation for his or her interlocutor. When they meet up around the same place of intention, they are said to have reached a commitment.

For a dialogue to take place at least one joint commitment is required for the speakers to be able to understand each other. These commitments motivate the clarifications and confirmations always present in everyday conversations. Note that these confirmations are often based on non-verbal communication, such as a face gesture or hand movement. But in phone conversations the speaker must frequently ask for agreement (either explicitly such as "aha?" or based on prosodic information) since he cannot see his interlocutor. If the speaker perceives that his or her

interlocutor does not understand him, he or she will reinforce the communication by repeating the idea. In such cases, the interlocutor should change the plane of interpretation slightly.

Apart from this, there is a weak point concerning the correct interpretation of the communication. Actually, there is very little (if any) theoretical foundation. However, recent research points to modelling the commitment, like theoretical models of joint action, for example (Clark, 1996). The joint-action model is based on the premise that a dialogue is fed by both talking entities and that they have at least one common reference point, a commitment. These joints have to be kept 'alive' by both entities with an acceptable level of certainty and efficiency, so they can refer to them. Therefore, they will need to ask for clarification to stop it falling too low, or to confirm or make enforcement utterances to let the interlocutor know that there is no need for redundant speech at that point.

7.1. TRAINS and TRIPS

The TRAINS System is an interactive planning assistant in a simplified transportation domain. For this particular sort of interaction, the user does not need previous training. Furthermore, he or she should utter what he or she wants, without constraints (Ferguson et al., 1996). Although it is not strictly speaking based on a joint-action model, it overtakes the plan-based approach by introducing the concept of 'obligation', which seems to be near enough to Cohen's 'commitments'.

The TRAINS family is a full series of progressively improved prototypes, which have added robustness, module independence in the architecture, and domain knowledge for constructing better plans. It is made up of a set of independent modules connected by message passing. Since the 96 version, the messages are expressed in KQML (Knowledge Query and Manipulation Language) (Ferguson and Allen, 1998), which defines a standard message structure for knowledge handling.

Processing can be divided into several phases: acquiring data, parsing to get speech acts, discourse management, planning and problem solving based on domain knowledge, discourse generation, natural language generation, and submission. For simplicity's sake, it can be seen as divided into four areas: acquisition/submission, NL parsing and generation, discourse management, and domain knowledge-based problem solving. These areas are clearly evinced in the system architecture, as shown in Fig. 11: the interface modules (acquisition/submission) at the top; NL processing and Dialogue Management in the middle; and the domain knowledge reasoning and problem-solver modules at the bottom.

For user input, both spoken speech and typed text are permitted. As for the output, the system generates spoken language, and displays text and maps. The input is processed in a robust parsing to obtain streams of speech acts. These semantic structures are input into the discourse manager.

Dialogue management is based on the discourse obligation premises. An 'obligation' is a line of action that cannot be omitted. An obligation arises from the acceptance of an offer or by following a command or a request. If the obligation consists

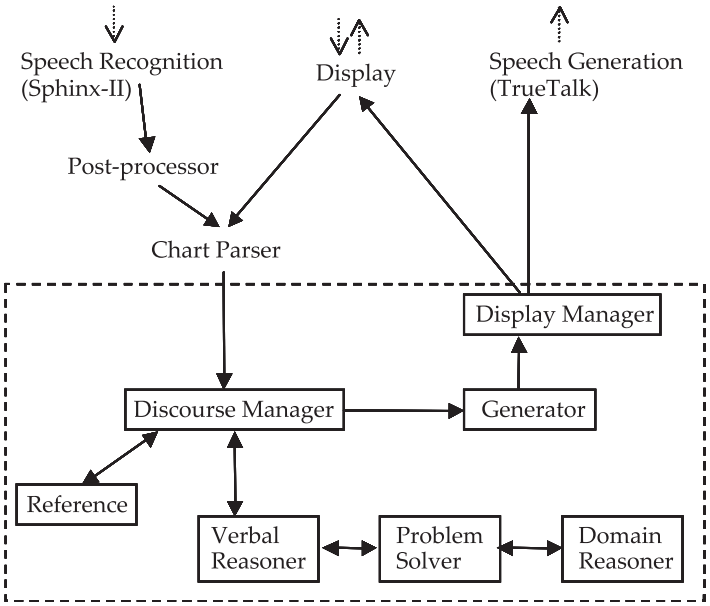

Fig. 11. Schematic architecture of TRAINS-96.

of saying something, it will be called 'discourse obligation' (Traum and Allen, 1994). This theory is implemented in TRAINS through two stacks, storing each interlocutor's pending obligations (which they have undertaken beforehand). Thus, this system observes system and user points of view (at least, it handles beliefs about user aims, plans, etc.).

The main components for discourse management in TRAINS should describe:

- Context manager: handles context information.
- Reference: solves anaphora.
- User Model: contains all there is to know about the user affecting the interaction (including intentions).
- World Model: system view of external world.
- Self Model: system view of its own beliefs and intentions.
- Display Model: system view of what is in the display (deictic references based on what is actually being displayed can even be processed).
- Prince: verbal reasoner for disambiguating speech acts, activating and communicating with the problem solver, and controlling the process.
- Actualisation: controls discourse and output generation.

Finally, if necessary, the problem will end up in the problem solver for planning generation. It should be noted that this is a mixed-initiative planning system. It does not just construct a sequence of operations for achieving a goal from a starting point, because the determination of which aspects of the world are relevant to the plan is actually a dynamic process. Therefore, it will firstly construct and explore the plan, and then co-operate with the user to reach the plan (by criticism, correction, and/or acceptance) (Ferguson et al., 1996).

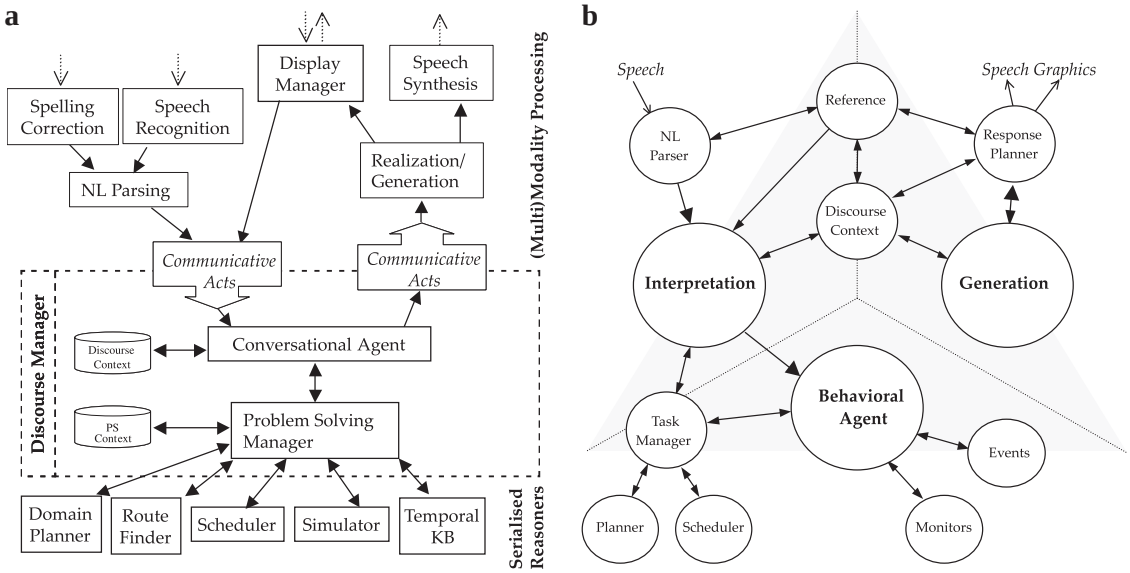

Fig. 12. (a) TRIPS Architecture. (b) Functional Arch. of TRIPS.

TRIPS (The Rochester Interactive Planning System) is built upon the same basis as TRAINS, but was made for more complex domains, more complex plans, and also improves collaborative problem solving (Ferguson and Allen, 1998). Its architecture (a more consolidated version of TRAINS') could be divided into three areas according to the function of the modules (as shown in Fig. 12a). These areas are modality processing, dialogue management, and specialised reasoning.

As for the functional point of view (Allen et al., 2000), TRIPS mainly relies on the following components (see Fig. 12b):

- Discourse Context: maintains everything, from current entities to discourse history, including details of preceding utterances, discourse obligations, etc.
- Abstract Problem-Solving Model: a set of actions that can be carried out on problem-solving objects.
- Task Manager: fixes the domain and how operations should be carried out.
- Interpretation Manager: parses user utterances to get speech acts.
- Generation Manager: joins goals and discourse obligations to construct plans.
- Behavioural Agent: defines system behaviour based on the interpretation of user acts, system goals and obligations, and occasionally some events.

To make this approach domain independent, the Task Model (Blaylock et al., 2003) is introduced. This concept will be used within this architecture as a container for domain-specific specialisations of problem-solving objects, such as:

- *objectives*: abstract goals in the domain;
- *recipes*: agent beliefs of how to attain an objective;
- *atomic actions*: executable actions within the domai;

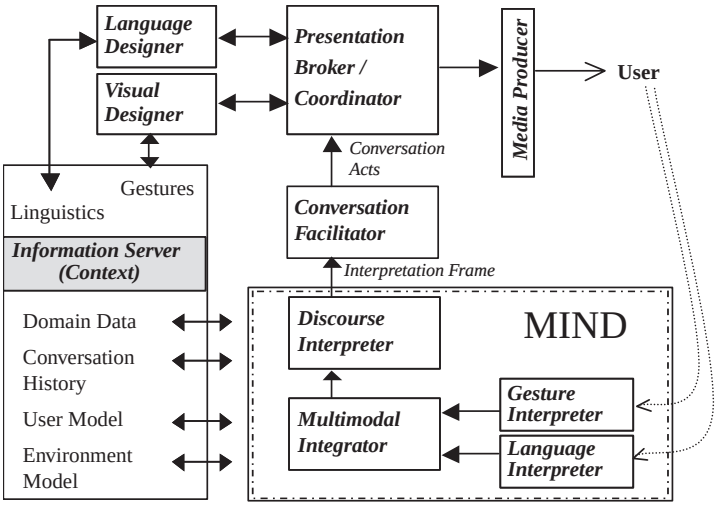

Fig. 13. Insights of the RIA system and the MIND.

- *objects/resources*: objects in the world, and objects used as resources in recipes;
- *situations*: (partial view of) the state of the world.

7.2. IBM Labs' MIND (Multimodal Interpretation for Natural Dialog)

This system includes the most applauded features of the previously released NLSA (multimodality, architecture, semantic structures) (Chai et al., 2001). This system was based on three main components: the presentation manager (for interface information acquisition and semantic interpretation), the dialogue manager (for discourse analysis based on domain corpus), and the action manager (for problem solving). But the discourse level analysis is more evolved in MIND (Chai et al., 2003), mainly based on Grosz and Sidner's theories of intention and attention. This approach is embedded in the Responsive Information Architect system, whose cognitive architecture for intentional multimodal interpretation can be seen in Fig. 13.

It makes a distinction between two main discourse elements: the unit (intention partially developed within a turn) and the segment (full intention, developed across at least one and usually several turns). The segment will be made up of five attributes: Intention, Attention, Initiator, Addressee, and State.

The discourse relations observed are of two types: structural relations (intention-subintention structures), and transitional relations (transitions between conversation segments and between conversation units; they can be further classed as two types: intention switch and attention switch, respectively). These intentional structures (segments) will help to handle the dialogue structure and achieve coherent dialogues.

8. Conclusions

The research in 'Natural Interaction' attempts to reach a system performing as a person when interacting. This approach is adequate to certain interaction domains

and users (technologically untrained), but other domains and systems will find other approaches more advantageous.

Specialised agents are needed to achieve Natural Interaction. It is hard to get an interaction agent able to perform any complete interaction within an interaction domain. Hence, the knowledge of 'how to interact' should be as far as possible separated from the rest of the (domain-dependent) knowledge. The proposed architecture is a step in this direction.

In addition, there is often an overlapping of interaction domains pointing to knowledge sharing as the solution to develop new agents easily with increased flexibility.

We presented a cognitive architecture for supporting an advanced HCI model, as well as an approach for evaluation. A first version of the cognitive architecture presented was implemented in a virtual assistant for e-commerce during the ADVICE project (IST-1999-11305), and an enhanced version was implemented within de VIP-Advisor Project (IST-2001-32440). This later version includes an implementation of the Threads Dialogue Model, with a 'common ground' modelling and commitment handling through interaction. It also encloses Session, Task, and Emotional Models, and Ontology. Completing the interaction system, it has several interface components (Natural Language Processing, voice recognising and synthesis, graphical user interface, and a 3D character), and External Agents for providing assessment and concept definitions in the Risk Management and Assurances domain.

The Threads Model includes, among other features, knowledge structures and reasoning that makes attuning the user and the system possible, thus enhancing interaction (making it easier and more useful, yet eventually longer). Apart from improvements in this line of work, the evolution of the model points to overlapped-turns interaction, and multi-participant interaction.

Acknowledgements

We thank the ISYS research group at the Technical University of Madrid and the LABDA research group at the Carlos III University of Madrid for their support during the design and development of the interaction agent, and everyone involved in the ADVICE (IST 1999-11305), VIP-ADVISOR (IST 2001-32440), and recent IntegraTV4All (FIT-350301-2004-2) projects for their hard work.

References

Allen, J., Ferguson, G., Stent, A., 2000 An Architecture for more Realistic Conversational Systems. IUI'00, Santa Fe.

Amores, J.G., Quesada, J.F., 2001. Dialogue moves for natural command languages. Procesamiento del enguaje Natural 27, 81 88.

Austin, J.L., 1962. How To do Things with Words. Oxford Univ. Press, 1975.

Bernsen, N.O., 2000. What is Natural Interactivity? In Procs. of WS: From Spoken Dialogue to Full Natural Interactive Dialogue. Theory, Empirical Analysis and Eval., pp. 34-37. Ed. L. Dybkjær. Second Int. Conf. on Language Resources and Evaluation (LREC'2000).

Bernsen, N.O., Dybkjaer, H., Dybkjaer, L., 1998. Designing Interactive Speech Systems: From First Ideas to User Testing. Springer.

Blaylock, N., Allen, J., Ferguson, G., 2003. Managing Communicative Intentions with Collaborative Problem Solving. In: van Kuppevelt, Jan, Smith, Ronnie W. (Eds.), Current and New Directions in Discourse and Dialogue, © 2003 Kluwer Academic Publishers. pp. 63 84.

Calle, F.J., 2004. Interacción Natural Mediante Procesamiento Intencional: el Modelo de Hilos en Diálogos. PhD Thesis. Technical University of Madrid.

Chai, J., Budzikowska, M., Horvath, V., Nicolov, N., Kambhatla, N., Zadrozny, W., 2001, Natural Language Sales Assistant A Web-based Dialog System for Online Sales. In: Proceedings of the 13th Innovative Application of Artificial Intelligence Conference (IAAI2001), Seattle, pp. 19 26.

Chai, J., Pan, S., Zhou, M., 2003. MIND: a context-based multimodal interpretation framework in conversational systems. In: Bernsen, O., Dybkjaer, L., van Kuppevelt, J. (Eds.), Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems. Kluwer Academic Publishers..

The Ciao Prolog Development System WWW Site. © 2004. The Computational Logic, Languages, Implementation and Parallelism Laboratory. <http://www.clip.dia.fi.upm.es/Software/Ciao/index.html>.

Clark, H.H. Using Language. © 1996, Cambridge University Press.

Cohen, P.R., Levesque, H.J., 1991. Teamwork, Nous 25(4), Special Issue on Cognitive Science and Artificial Intelligence, pp. 487 512.

Ferguson, G.F., Allen, J.F., 1998. TRIPS: An Integrated Intelligent Problem-Solving Assistant. In: Proceedings of the 15th National Conference on Artificial Intelligence, pp. 567 572.

Ferguson, G.F., Allen, J.F., Miller, B.W., Ringger, E.K., 1996. The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant. TRAINS Technical Note 96-5. Rochester, New York.

FIPA Communicative Act Library Specification. Foundation for Intelligent Physical Agents, 2000 2002. <http://www.fipa.org/specs/fipa00037/SC00037J.pdf>.

Garcia-Serrano, A., Calle-Gómez, J., 2002. A cognitive Architecture for the design of an Interaction Agent. In: Klusch, M., Ossowski, S., Shehory, O. (Eds.), Cooperative Information Agents VI. Lecture Notes in Artificial Intelligence. Springer, pp. 82 89.

Gee, J.P., 1999. Introduction to Discourse Analysis. Routledge.

Grosz, B., Sidner, C., 1986. Attention, Intention, and the Structure of Discourse. Computational Linguistics vol. 12, no. 3, pp. 175 204.

The Java Technology webpage, Sun Developer Network. ©1994 2006, Sun Microsystems, Inc. <http://java.sun.com/>.

Kowtko, J.C., Isard, S.D., Doherty, G.M., 1993. Conversational Games Within Dialogue, Research P. 31, Human Comm. Research Centre, Edinburgh.

Levin, J.A., Moore, J.A., 1977. Dialogue games: metacommunication strategies for natural language interaction. Cognitive Science 1 (4), 395 420.

Levinson, S.C., 1983. Pragmatics. Cambridge University Press.

Martínez P., García Serrano A., Calle J., Rodrigo-Aguado, L., 2001. An agent-based design for a NL-interaction for intelligent assistance in the e-commerce scenario. In: IEEE International Workshop on Natural Language Processing and Knowledge Engineering.

Traum, D.R., Allen, J.F., 1994. Discourse Obligations in Dialogue Processing. In: Proceedings of ACL.